Enhanced Probabilistic Packet Marking for IP Traceback

Joseph Gomes Fanchun Jin Hongsik Choi Heyong-Ah Choi

Abstract—

The recent proliferation of Denial of service attacks (DOS) has caused a lot of concerns in the security community. This increase can be mainly attributed to the lack of accountability on the attacker’s part. The use of ‘forged’ IP source addresses during DOS attacks render this information retrieved from the attacking packets useless as a means to trace the attack back to its source. Currently there is no single scheme that is widely used in tracing back a DOS attack. The current approaches are liable to either increasing the network overhead, processing done at the routers or processing done at the victim node. One of the approaches that has garnered a lot of attention recently is the probabilistic packet marking (PPM) approach. In this paper we introduce an extension of this probabilistic packet marking scheme that significantly reduces the number of packets required to reconstruct the attacking path.

I. INTRODUCTION

In the recent past the number of DOS attacks has gone up significantly. There is no current technology that can hold the attackers accountable. The attackers can remain anonymous and do not have to face the risk of getting caught and punished. In a study carried out by Moore et al. [1], more than 12,000 DOS attacks were observed towards 5,000 different hosts belonging to 2,000 different organizations within a period of only three weeks. The intensity of the packet rate among these attacks ranged from 50 packets per second (pps) to 650,000 pps, with a mean around 1,000 pps. The duration ranged from 1 minute to several days, with a mean around 10 minutes. Thus a typical DOS attack would consist of around 600,000 packets.

However there has not been any effective way to reduce the onset of DOS attacks. The main reason behind this is the incapability of identifying the attackers and holding them responsible for their actions. In the current version of IP protocol the sender itself adds the source IP address to the packet and there is no mechanism built into the routing protocols that verifies the authenticity of this source address. As a result a sender with malicious intention can easily get away with using a fake IP address as the source address. The only defenses available today are in the form of access control lists, firewalls and IDS systems. However these methods only mitigate the effect of the DOS attacks on the victim but they do not discourage the attackers. Moreover even firewalls can be disabled by packet rates in excess of 14000 packets per second according to [1]. The notion of holding attackers accountable was first introduced in [2]. Since then there have been numerous ideas and approaches proposed that connect the attacks back to the malicious parties involved. Edge sampling using probabilistic packet marking has recently caught a lot of attention as a means of tracing the attack back to its origin. Solving the trace back problem seems to be the only way which could lead to the solution to the accountability issue. The mere possibility of getting caught will restrain most of the attackers. The edge sampling scheme by Savage et. al. [3] makes a big leap towards making trace back possible. However it is not very effective in solving the real time traceback problem as a victim needs to receive a large number of packets from an attacker before it can deduce the source of the attack [4]. In this paper we propose an enhanced version of their scheme that will enable the victim to deduce the attack path within receiving a very small number of packets from the attacker. We use the number of packets required from the attacker by the victim to reconstruct the path as the primary metric for the effectiveness of an algorithm. Throughout the rest of the paper we will denote this number by N. We also show that by increasing the number of marks for each packet we reduce N even further.

II. RELATED WORK

There are many approaches to the IP traceback problem. IP traceback using probabilistic packet marking has received a lot of attention lately. In this section we will go over some of the packet marking schemes that have been proposed before and are relevant as background knowledge for our scheme.

A. Node append

This is the simplest of all marking algorithms. In this algorithm all the routers in the path from the source to destination append their IP address at the end of packet. When the packet arrives at the destination it contains an ordered list of all the nodes the packet has passed through. However it has serious limitations in that it incurs heavy router processing overhead and increases packet size by a significant amount. It may even cause IP fragmentation of

Joseph Gomes: George Washington University, Washington, DC.
Fanchun Jin: George Washington University, Washington, DC.
Hongsik Choi: Virginia Commonwealth University, Richmond, VA.
Heyong-Ah Choi: George Washington University, Washington, DC.

the packets whenever the packet length goes over the MTU size as a result of all the appendage and cause problems for services like path MTU discovery.

B. Node Sampling

This algorithm solves the problem of processing overhead and increased packet size introduced by the Node Append algorithm. Here, instead of having all the router addresses in one packet, they are distributed among all the packets that reach the destination. Each packet contains at most one router address. When a router receives a packet it inserts its address into the packet with some probability \( p \). After receiving a sufficient number of packets the victim can expect to have received router marks from all the routers in the path. The problem with this scheme lies in figuring out the order of the routers in the path. This problem can be overcome by considering the relative number of marks received from each router. As each packet contains at most one router mark, allowing routers to overwrite marks made by previous routers in the path leads to a monotonically decreasing probability of receiving marked packets from a router the further away it is from the destination. In other words, theoretically the victim will receive more marked packets from a router at distance \( i \) than a router at distance \( i + 1 \). Thus by looking at the number of marked packets from each router the victim can deduce their relative order in the path. However, due to the randomness inherent in the algorithm it requires the victim to receive a large number of marked packets from each router before it can deduce the order with confidence. Moreover it will run into problems if there are multiple attackers at the same distance, as the victim will receive similar numbers of markings from multiple attack paths.

C. Edge Sampling

This algorithm overcomes node ordering problem by marking edge information in each packet instead of node information. Each marking contains the start address representing the starting node of the edge, the end address representing the end node of the edge and a distance field representing the distance of that edge sample from the destination. Whenever a router decides to mark it writes its own address in the start address and sets the distance to zero, regardless of whether the start address is empty or not. If a router decides not to mark, it puts its own address in the end field given the distance is zero. The zero distance is an indication that the previous router has written its address in the start field. Additionally when the router decides not to mark it increments the distance field regardless of whether it is the end node of the edge sample or not. The distance field helps the victim node in ordering the edge samples. It also prevents the attacker from pretending to be one of the intermediate routers. If a fake mark by the attacker reaches the victim without being overwritten, its distance will always be greater than those of the legitimately marked packets.

The probability that a packet will contain an edge sample is at least \( dp(1 - p)^{d-1} \) where \( d \) denotes the attack path length or the number of edges between the attacker and the victim. Since the number of trials required to select each of the different edge samples from \( d \) equally probable (using the lower bound) edge samples is \( d(ln(d) + O(1)) \) using the ‘coupon collector’ problem, the expected value of \( N \) is bounded by the equation

\[
E(N) < \frac{\ln(d)}{dp(1 - p)^{d-1}}
\]

The problem with this algorithm is that it requires an additional 72 bytes (2 \( \times \) 32 bytes for two IP addresses and 8 bytes for the distance) per packet to insert the mark. Savage et al. addressed this problem in [5] by introducing an encoding scheme that only uses the seldom used identification field to store the edge marks. Their scheme first uses XORing to reduce the 64 bytes of address to a 32 byte edge-id, then splits these 32 bytes into multiple fragments to take care of the space problem in the IP header. Each packet can carry at most a single fragment, its associated distance and its position within the XORed edge-id. However fragments from different edge-ids could be equal and cause ambiguity in reconstructing the edge-ids. In order to overcome this collision problem they bit-interleave the hashed value of a router address with the router address before using XOR. This doubles the size of the edge-ids and consequently increases the number of fragments required per edge sample. The expected value of \( N \) for this algorithm is given by:

\[
E(N) < \frac{k \cdot \ln(kd)}{dp(1 - p)^{d-1}}
\]

where \( k \) is the number of fragments needed to encode an edge sample. For \( d = 10 \) and \( p = \frac{1}{2} \), \( E(N) < 1300 \). However the victim will need around 2600 packets to reconstruct the path with 99% confidence. Later we will show that we were able to reduce this number by a significant amount using our scheme.

D. Others

There have been many other proposals for solving the IP traceback problem. Bellovin proposed that routers probabilistically create ICMP packets with traceback information in [5]. However with his suggested marking probability the convergence time for this scheme is very long.

Source Path Isolation Engine (SPIE) [6] is a logging based traceback scheme. Instead of storing each packet SPIE computes and stores only the digest of a small portion of a packet which includes the invariant portion of the IP header (16 bytes) and the first 8 bytes of payload. Each
Once the victim has received at least $\varepsilon$ is chosen and the mark $i$ is added to the packet it recomputes the $k$ digests and verifies if each of those $k$ index positions are set or not. The packet is deemed to not have passed through the router, if any one of the bits are not set. The problem with SPIE is that the victim cannot reconstruct the path on its own. It has to issue a reconstruction request to the SPIE Traceback Manager (STM) which in turn propagates and distributes the requests to the SPIE Collection and Reduction Agents (SCAR) who are responsible for reconstructing attack graphs for different areas of the network. Each router then has to perform table lookups to verify if the packet was stored. As a result the reconstruction process is very time consuming and resource intense.

The Advanced and Authenticated Marking Scheme (AMS) proposed by Song and Perrig [8] also uses hash functions to create the marks. However their marks are not stored in a digest table; rather they are inscribed in the packets. The AMS scheme uses a set of 8 different hash functions $H_i : 1 \leq i \leq 8$ for mark creation. A random $i$ is chosen and the mark $i \parallel H_i(e)$ is added to the packet where $e$ represents the edge the packet is going through. Once the victim has received at least $k$ out of the 8 possible mark values for a particular edge it considers that edge to be a part of the attacking path. $k$ is a changeable parameter and the suggested range for $k$ is between 6 and 8. The drawback of this algorithm lies in the assumption that the victim knows the universal tree $U$, which is quite an unrealistic and impractical assumption.

Song and Xu were the first to propose a filtering scheme[9] that takes advantage of a traceback algorithm to gather intelligence and creates traffic filters based on the gathered information. In this scheme routers probabilistically mark the packets with two different types of marks: signal mark and data mark. Signal marks carry the same information as any IP traceback scheme and constitute around 5% of the total marks. However due to space constraints in a packet the signal marks are broken into multiple pieces and each of these separate pieces is encoded in a separate packet. As a result a specific piece contained in a packet does not contain enough information about the edge to base the filtering scheme upon. Hence the need for data marks. The data marks are 16 bit hashed values of a network edge and thus carry the whole identity of the edge, rather than a fragment of the edge. The edge information contained in these data marks along with the attack graph reconstructed from the signal marks together help decide in choosing the filtering criteria and filtering probabilities. The filtering criteria and probabilities are sent to the perimeter routers that discriminate the packets to be infected or clean based on the criteria and filters them according to the probabilities. This scheme suffers from affecting legitimate traffic as it is filtering traffic based on only one edge information rather than the whole path.

Yaar et al. [10] took this idea of filtering based on packet marks and tried to reduce the effects of filtering on legitimate traffic. In their path identification mechanism PI, a packet is deterministically and distinctively marked by each router in the path. Each router uses the last $n$ bits of the hashed value of its IP address as its mark. The IP Identification field is broken into $\lfloor 16/n \rfloor$ parts. Each router uses $TTL$ modulo $\lfloor 16/n \rfloor$ as its relative position in the Identification field to add the mark. As a result all packets going through the same path carries the same mark and packets going through different paths have the same mark with a very low probability. Consequently each mark can be thought of as a distinctive identifier for the path the packet has traversed, making it a very good candidate for a filtering basis. Once the victim separates the malicious packets it can use the markings in those packets to filter out malicious traffic.

### III. Enhanced Marking Scheme

#### A. Overview

Our scheme is very similar to the edge sampling algorithm described in [3]. The two algorithms differ in the number of marks that are added per packet and in the way the marks are created. Firstly, the edge sampling algorithm adds one edge marking per packet. However in our algorithm we can add at most $k$ edge markings. Secondly, the edge sampling algorithm adds the edge marking within the packet, whereas in our case the first router that decides to mark the packet has to create a new IP packet that contains the marking information. We call these newly generated packets ‘marking packets’. The marking packets only copy the IP header portion of the original packet (Figures 2 and 3). If a marking packet was already generated for a particular data packet by an upstream router, no subsequent routers will generate a new marking packet for it. The original edge sampling algorithm was constricted by space constraints as it was adding the marks within the data packets itself. In fact it needed to fragment the edge marks into multiple pieces and embedded each fragment in a separate packet. As a result it required multiple packets to carry a single edge mark. However using totally new packets for the edge marks gives us the opportunity to add as many edge marks per packet as we want. We denote the maximum number of edge marks that can be added per packet to be $k$. In the original marking algorithm if multiple routers decided to mark the same packet then the latter routers would have to overwrite the previous marks. Since the further away a router is from the victim, the higher is
the probability for its mark to be overwritten by a subsequent router. Consequently, the convergence time for the path reconstruction problem is heavily dominated by the time it takes for the furthest edge from the victim to reach the victim. On the other hand if subsequent routers are not allowed to overwrite existing edge marks then the opposite happens and the victim would have to wait a long period of time before it receives edge marks for the links that are closest to itself. Our algorithm solves this problem by removing the constraint of having only one edge mark per packet. We think that allowing at most k edge marks per packet will significantly reduce the convergence time for the path reconstruction problem.

IP header of a data packet to indicate whether a marking packet has been generated for a data packet or not. We also make use of the Identification and Don’t fragment fields in the IP header as according to [11] less than 0.5% of all packets in the Internet are fragmented. The ‘Don’t Fragment’ flag is used to indicate whether the packet is an original data packet or a generated ‘marking packet’. Distinguishing data packets from ‘marking packet’s prevents routers from generating marking packets for other marking packets. The router that first creates a marking packet sets the unused flag in the original packet to let the subsequent routers know that a corresponding marking packet has already been generated for that packet. A router does not create any additional marking packet if this flag is set for a given packet. This introduces the problem of the attacker always setting this flag in the packets it sends and thus preventing the routers from ever creating marking packets. This can be dealt with by using a time-released authentication scheme similar to the one used in [8]. This authentication scheme uses message authentication codes (MAC) such as HMAC-MD5 with time-released key chains and can be adapted so it only requires the 16-bit IP Identification field for storage. The time-released key chain avoids having a separate private key between each pair of routers. The router \( R_i \) that sets the unused flag in the original packet computes a MAC using its key \( K_{t,i} \) for that time interval and adds it to the IP Identification field. \( R_i \) will then disclose the key \( K_{t,i} \) during time interval \( t + 1 \). The keys are computed using a one way hash function in such a way that given the key for the current time interval anybody can compute the keys for the previous time intervals but not the other way around. When a router \( R_j \) finds that a data packet has been marked already, it downloads the keys for the upstream routers, determines the sending time interval of the packet, computes the key for that time interval and verifies the authenticity of the mark. Note that router \( R_j \) will have to wait for the key after receiving the packet due to the key disclosure delay. However this is going to be very rare (when \( p \) is small) since the authentication will only take place when the router probabilistically decides to mark and the marked flag of the data packet is set. Additionally the situation can be further improved by choosing the length of a time interval to be much smaller than the average transmission delay of a packet between routers. In that case the routers could use keys that have been received earlier but are of a later time interval than \( t \). The marking information pertaining to a single edge consists of a start address, end address and a distance field. The router that initiates an edge marking adds its address to the start field and initializes the distance to 0. If the next router decides not to mark and if the distance is zero, it puts its address in the end field and increments the edge count. The distance value of zero is an indication that the mark was initiated by the previous router. On the other

![Fig. 1. Data Packet: encoding marked flag in the unused flag](image1)

![Fig. 2. Marking Packet: encoding the edgecount in the identification field](image2)

![Fig. 3. IP Payload for Marking Packet](image3)
hand if the next router decides to mark, it will overwrite its previous router’s mark. The edge count in our algorithm represents the number of edges that have been completely marked. The identification field (Figure 2) of the marking packet is used to store the edge count. The marking packets are very small in size. We need 72 bits for the start address, end address and distance of each edge mark. Depending on k, the size of each marking packets will be 20 + 72 × k bytes. An edge mark is considered complete and hence the way that this never happens. Finally if a router decides can be set according to the marking probability in such a 1. This reduction is done because a complete edge mark is being replaced by an incomplete edge mark. However k can be set according to the marking probability in such a way that this never happens. Finally if a router decides not to mark then it will first complete any incomplete edge marks initiated by its previous router and increment the distance for each of the complete edge marks.

B. Algorithm

Marking algorithm at router R
$array indices are 1-based$

for each packet w
let x be a random number between [0..1]
if x < p then
if w is a marking packet then
let $R \rightarrow w$.edge[edgecount + 1].start
write 0 into $w$.edge[edgecount + 1].distance
else
let i be a random integer from [0..k]
write $R \rightarrow w$.edge[i].start
write 0 into $w$.edge[i].distance
reduce edgecount by 1
else
if no marking packet was generated for w then
set w.marked to true
create marking packet m for w
write $R \rightarrow m$.edge[1].start
write 0 into $m$.edge[1].distance
else
for i = 0 to edgecount
if $w$.edge[i + 1].start is not empty and
$w$.edge[i + 1].distance = 0 then
write $R \rightarrow w$.edge[i].end
increment edgecount by 1
increment $w$.edge[i].distance by 1

Path reconstruction procedure

let $G$ be a tree with root v
let edges in $G$ be tuples (start, end, distance)
for each marking packet m
for $k = 1$ to edgecount
if $w$.edge[i].distance = 0 then
insert edge($w$.edge[i].start, v, 0) into G
else
insert edge($w$.edge[i].start, $w$.edge[i].end, $w$.edge[i].distance) into G
remove any edge $(x, y, d)$ with $d \neq$ distance from x to v in G
extract path ($R_1..R_j$) by enumerating acyclic paths in G

C. Analysis

In this section we provide an analysis for finding the expected value of $N$.

Theorem 1: Let $k$ be the size of buffer for edge sampling, $d$ be the attack path length, where $d \geq k \geq 2$, and $p$ denote the probability by which a router marks a packet. For $1 \leq i \leq d$, $e_i$ is the edge sample $i$ hops away from the router. Let $P_i$ denote the probability that the edge sample $e_i$ can reach the victim without being overwritten. Then the probability of a packet containing an edge sample from the furthest edge $e_1$ in the attack path is given by:

$$P_1 = p(1 - \sum_{i=k}^{d-1}(C_i^{d-1}p^{i}(1-p)^{d-1-i}(1-(1/(k)^{i-k+1})))$$

Proof: In our algorithm, without losing any generality we can assume that instead of routers marking the packets the edges mark the packets with probability $p$. After a packet is marked by the first edge $e_1$ in the attack path, it will go through subsequent $d - 1$ edges, and an edge $e_i$ will add its label to the packet with probability $p$. The probability that a packet marked by $e_1$ is also marked by $i$ subsequent edges is $C_i^{d-1}p^{i}(1-p)^{d-1-i}$. However if the buffer for edge labeling is full(i.e. $i \geq k$), the edge $e_i$ will overwrite the label of the edge $e_1$ with probability $1/k$. Therefore the probability that the mark of $e_1$ is not overwritten by any of these $i - k + 1$ edges is given by $((1-1/k)^{i-k+1})$. This implies that an edge sample for $e_1$ can be overwritten by a probability of

$$\sum_{i=k}^{d-1}(C_i^{d-1}p^{i}(1-p)^{d-1-i}(1-(1/(k)^{i-k+1})))$$

So the probability that a packet carries a mark from $e_1$ is given by

$$p(1 - \sum_{i=k}^{d-1}(C_i^{d-1}p^{i}(1-p)^{d-1-i}(1-(1/(k)^{i-k+1})))$$

Now since for any $i > 1$ $P_i < P_1$, we can conservatively assume for all the edge samples the probability of reaching
the victim is equal to that of the furthest edge. Therefore a given packet will have a mark from some edge with probability $dP_1$. As per the coupon collector problem, we need $d(ln(d) + O(1))$ trials to select all $d$ different types of edge marks if each packet always contains exactly 1 mark. However as packets could have at most $k$ marks, we could get all of the different types of edge marks in fewer trials. Therefore, the expected value of $N$ is given by

$$E(N) < \frac{d(ln(d) + O(1))}{dP_1} \approx \frac{ln(d)}{P_1}$$

$$\approx \frac{ln(c)}{P_1}$$

Fig. 4. Comparison of simulation results with expected results from equation: $d=15$, $k=2$, certainty level=50%

D. Certainty analysis

We want to stress on the issue of being able to reconstruct the path with a certain confidence. The analysis done in section 3.3 only gives us the expected value of $N$ for a single attack path during a single attacking incident. However it is not sufficient to guarantee that the victim will be able to reconstruct any attacking path; in fact the victim will not be able to reconstruct a complete path around half of the time. This is why we are more interested in the value of $N$ that guarantees path reconstruction with 99% certainty. Based on the coupon collector problem we need $d(ln(d) + ln(c)) = d(ln(cd))$ trials to select one of each of $d$ equally probable coupons with confidence probability $(1 - \frac{1}{c})$. For example, if confidence probability=0.95, $c = 20$. However as packets could have at most $k$ marks we can approximate the value of $N$ that guarantees path reconstruction with confidence probability $(1 - \frac{1}{k})$ as:

$$E(N) < \frac{d(ln(cd))}{dP_1} \approx \frac{ln(cd)}{P_1}$$

Later we will show some results obtained from this equation and compare them to our simulation results.

Fig. 5. Comparison of simulation results with expected results from equation: $d=15$, $k=2$, certainty level=99%

Fig. 6. Simulation results of $N$ vs. $k$ for the 99% certainty case: $d=15$

IV. Simulation results

We simulated our algorithm to find out the expected value of $N$ for different marking probabilities using different values of $k$. We also found out the number of packets required to reconstruct the path with 95% and 99% certainty. In our simulation we assume that there will be no packet drops. For the results shown here the attack path length $d$ was set to 15. Marking probabilities for the routers ranged from 1/15 to 1/1000. For each marking probability we ran 100,000 trials to get the results. In Figures 4 and 5 we can see that our simulation results conform very well with the expected result from the equations provided in section 3.4. From our results we have observed that the required number of packets can be reduced by as much
V. Network Overhead

In our scheme the routers generate additional packets to create their marks. This causes additional network overhead. However if a marking packet has already been generated by a previous router for a given packet no subsequent routers create additional marking packets for that packet. The routers can check the unused IP flag of a given packet to verify whether a marking packet has been generated or not.

The additional number of packets generated by our scheme is proportional to the marking probability assigned at the routers. For example (Figure 8) if $d = 10, p = \frac{1}{50}$, and $k = 2$ our scheme will generate around 25% additional packets in general. When $p$ is decreased to $\frac{1}{100}$, additional network overhead decreases to 13%. However as we have mentioned earlier this should not be an issue in cases where network bandwidth is under-utilized. It should also be noted that marking packets do not carry payload of the corresponding data packets. Furthermore, a messaging protocol could be established between routers and end nodes, where the end nodes can trigger the marking procedure at the routers by sending authenticated messages. This way the additional network overhead will be experienced only when an end victim is under attack.

VI. IPv6 Implementation

A direct implementation of our scheme is not possible in IPv6 due to its differences with IPv4 [12]. IPv6 does not have the rarely used fields such as Identification, TOS, Fragmentation flags in the header. The most appropriate place in an IPv6 packet to put the edge mark would be in the Hop by Hop extension header. A new hop by hop option could be introduced that would allow the routers to add their marks in the Hop by Hop extension header.
However this would introduce a fragmentation problem in the intermediate routers since IPv6 does not allow router to perform fragmentation. In IPv6 fragmentation is only allowed at the source. A router can check whether adding a mark would cause fragmentation before doing so. Even that does not help since the attacker can always set the packet sizes in such a way that the router will never be able to add any mark without causing fragmentation.

VII. Conclusion

In this paper we have proposed a new approach using edge sampling based on probabilistic packet marking. Our algorithm can reconstruct the attack path using significantly fewer number of packets than the other approaches at the expense of additional network overhead. We have also shown that increasing $k$ by as little as 1 (from 1 to 2), which represents the number of edge marks each marking packet can hold, we can significantly reduce the number of packets required both in expectation and in case of 99% confidence.

References